QUANTICOL

A Quantitative Approach to Management and Design of Collective and Adaptive Behaviours

quantiCOL

http://www.quanticol.eu

D3.2

Scalability beyond population size and quantitative product family engineering

Revision: 0.0; September 29, 2015

Author(s): Mieke Massink (Ed.) (CNR), Maurice ter Beek (CNR), Cheng Feng (UEDIN), Jane Hillston (UEDIN), Mirco Tribastone (SOTON, IMT)

Due date of deliverable: Month 30 (September 2015)

Actual submission date: Sept 30, 2015

Nature: R. Dissemination level: PU

Funding Scheme: Small or medium scale focused research project (STREP)
Topic: ICT-2011 9.10: FET-Proactive 'Fundamentals of Collective Adaptive Systems' (FOCAS)
Project number: 600708

Coordinator: Jane Hillston (UEDIN)
e-mail: Jane.Hillston@ed.ac.uk
Fax: +44 131 651 1426

Part. no.	Participant organisation name	Acronym	Country
1 (Coord.)	University of Edinburgh	UEDIN	UK
2	Consiglio Nazionale delle Ricerche – Istituto di Scienza e Tecnologie della Informazione "A. Faedo"	CNR	Italy
3	Ludwig-Maximilians-Universität München	LMU	Germany
4	Ecole Polytechnique Fédérale de Lausanne	EPFL	Switzerland
5	IMT Lucca	IMT	Italy
6	University of Southampton	SOTON	UK
7	Institut National de Recherche en Informatique et en Automatique	INRIA	France

COOPERATION

Executive summary

The main aim set out for Work Package 3 is the development of the theoretical foundations of novel, scalable and spatial formal analysis techniques and the underlying theories to support the design of large scale CAS. During the first year of the project we have developed several innovative analysis techniques that are highly scalable. Some of these are based on mean field approximation techniques, others involve statistical model checking. In both cases, the development of additional model reduction techniques is very important to further improve scalability of analysis, for example to reduce the number of ordinary differential equations (ODEs) that need to be solved or the number of populations that need to be considered. The description of the project's achievements concerning innovative model reduction techniques constitutes one of the two main parts of this deliverable. In particular this first part addresses: the development of suitable behavioural equivalence relations on ODEs to support model reduction and comparison; a method to identify and remove populations that have no significant impact on a measure of interest; and a method based on automatic moment-closure analysis of population CTMCs.

The second part addresses the project's achievements concerning the development of suitable extensions of a software product line engineering (SPLE) approach for CAS. In particular, family-based verification of behavioural aspects of CAS has been pursued. In family-based analysis, the system model covers both behaviour that is common to all products of the family as well as variation points used to differentiate among the individual products that can be derived from the family. This way, logical properties can be analysed at the family level using variability knowledge to deduce results for products, rather than having to generate and analyse each single product, which is often very costly and does not scale well. In this context, an SPLE modelling and analysis framework has been further developed and implemented in the Variability Model Checker VMC that provides family-based verification of qualitative state and action based properties in an on-the-fly fashion. Furthermore, an alternative approach is presented to reason about software product lines focussing on behavioural relations. To this aim the Variant Process Algebra VPA has been developed that has a family-based semantics in which variants can be explicitly labelled. Several quantitative extensions of variability analysis have been developed to handle variability in software performance models. Finally, a first proof-of-concept for feature-oriented *modular* verification has been developed and, for a restricted notion of coherent branching feature bisimulation, a minimisation algorithm has been developed and its correctness has been shown, as briefly discussed in this report.

Contents

1 Introduction

This deliverable reports on results achieved in the context of the first phase of Task 3.2 "Abstraction Techniques for Scalability Beyond Population Size" and Task 3.3 "Relating Local and Global System Views with Variability Analysis". Progress on Task 3.1 "Spatial Stochastic Logics and Scalable Verification" will be reported in a forthcoming Internal Report that will appear in month 36.

Task 3.2 is mainly concerned with finding suitable behavioural equivalence relations based on ordinary differential equations (ODEs). In the context of the QUANTICOL project collective adaptive systems (CAS) are considered to consist of a large number of heterogeneous entities with decentralised control and varying degrees of complex autonomous behaviour. There is an increasing tendency to model the behaviour of such systems by the solution of ODEs. The idea is that such a solution can be interpreted as the deterministic (also called *fluid* or *mean-field*) approximation[1] of a continuous time Markov chain (CTMC). Such CTMCs are also underlying process algebraic languages with Markovian semantics. ODE semantics of such languages define the behaviour of a concurrent program as a continuous trajectory representing the 'concentration' of processes in each of their local states. The solution of a set of ODEs is independent of the number of processes considered, providing very good scalability. However, clearly the number of differential equations that need to be solved grows linearly with the number of local states of the processes. Moreover, there may be multiple classes of such processes. Behavioural equivalences can be used to reduce the number of differential equations that need to be solved when analysing a system model. The main aim is to lift the notion of bisimulation to languages with ODE semantics. A central role in this work is played by the well-known theory of ODE lumpability: the solution to each ODE representing an equivalence class is equal at all time points to the sum of the solutions of the ODEs of the states in that equivalence class. Such an approach can also establish a bridge between fluid (ODE-based) semantics and hierarchical modelling of systems-of-systems. Improved scalability can also be obtained in other ways. One approach is inspired by previous work in combustion modelling and used to considerably reduce the running time of simulations required in statistical model-checking approaches[2] by identifying populations within the model which can be ignored or removed without having significant impact on identified target populations. Another approach exploits an automatic moment closure approach to describe the evolution of the first-order, the second-order, and the second-order joint moments of an arbitrary population CTMC model. Achievements in model reduction approaches are outlined in Sect. 2 of the current deliverable.

Task 3.3 is concerned with a product line engineering view of CAS. Product line engineering is a software engineering approach aiming at the development and maintenance of a family of software-intensive systems by the systematic reuse of components or subsystems from related products or earlier versions of a product. Important questions from a SPLE point of view concern the multi-objective optimisation of feature variability models with attributes that cater for quantitative constraints like cost and customer satisfaction. Although so far the focus in SPLE has mostly been on structural properties, in many cases feature variability has also a behavioural component. The latter can be both qualitative in nature, for example additional features that lead to new interactions between a user and a system – or between system components – and quantitative, for example increased costs or improved performance of system components. A further issue in SPLE of CAS is the multiplicity of similar components that the overall system may consist of and the variability in this multiplicity that may profoundly affect the performance and quality of service of the overall system. Furthermore, variability can also be found in the parameters of the system.

The outline of the Deliverable is as follows. Section 2 presents an overview of the achievements in the development of specific abstraction techniques to improve scalability of the analysis of population models. In Section 3 an overview is given of the work on variability analysis for CAS.

[1] We refer to Section 2 of Deliverable 1.1 "Multiscale modelling informed by smart grids" for a brief introduction to mean-field and fluid techniques.

[2] We refer to Section 4 of Deliverable 3.1 "Foundations of scalable verification for stochastic logics" for a brief introduction to statistical model-checking.

2 Abstraction Techniques for Scalability Beyond Population Size

This section presents an overview of the achievements in the development of specific abstraction techniques to improve scalability of the analysis of population models. First we briefly recall the state of the art before the start of the project and the objectives set for this task for the second reporting period. This is followed by an overview of the main achievements for this task.

2.1 State of the art/baseline before start of the project

With respect to equivalence relations based on ODEs, the state of the art in the context of formal models based on process algebra is [85], where the notion of *label equivalence* is presented for a fragment of PEPA with fluid semantics. This essentially captures isomorphic sequential processes that exhibit the same ODE trajectories when they are initialised with the same initial conditions. The relationship between formal languages and ODEs induced by their semantics has been studied also in other contexts, with complementary approaches. In [48] a model-order reduction technique is presented for κ [47], a rule-based language for chemical systems with mass-action semantics representing bindings between molecules in an explicit graph-based way. The aggregation method, called *fragmentation*, identifies a linear transformation of the state space yielding a subspace with a closed dynamics, i.e., whose ODEs depend only on the variables of that subspace. This usually gives a *covering* of the state space as opposed to a proper partition as in [85], since the same ODE variable may appear in more than one aggregate.

The combination of the notion of bisimulation and ODEs has been explored also by the control theory community, most notably in the work of Pappas and co-authors (e.g., [73, 62]) and Van der Schaft [78]. However, the setting is different. When studied for model reduction, they essentially deal with a state space representation with an explicit output map, e.g., the matrix C in the linear dynamical system $\dot{x} = Ax + Bu$, $y = Cx$. A bisimulation is thus related to unobservability subspaces (cf. [73, Section 8.1] and [78, Corollary 6.4]). By contrast, both [48] and [85] deal with nonlinear systems in the form $\dot{x} = A(x)$ (with A a nonlinear vector field) where reductions are state-space aggregations.

2.2 Objectives

With respect to equivalence relations for ODE, the main objectives set for this reporting period can be summarised as follows:

1. To generalise the notions of behavioural equivalence for process algebra by allowing aggregations at a finer granularity than whole sequential components (i.e., by relating individual states, that are the constituents of the underlying ODE systems.)

2. To consider other ODE symmetries than label-equivalence type relations. In particular, to study ODE counterparts of Markovian bisimulation, to identify classes of variables that can be self-consistently written as a single macro-variable that represents their sums.

3. To go beyond the rules of synchronisation of the process algebra presented in [85], capturing for instance semantics based on the law of mass action.

4. To relax the assumption of exactness, allowing approximate relations that provide bounds on the accuracy of the approximation.

With respect to other approaches to improve scalability of fluid approximation of CAS, the main objectives were to combine reduction techniques with model-checking approaches and to exploit automatic moment closure techniques to address higher moments of population CTMC models.

2.3 Achievements in the second reporting period

The achievements obtained during the reference period addressed in this deliverable concerning Task 3.2, from month 7 upto month 30, are described in detail in the following publications: [24], [20], [25], [23], [16], [17], [18], [19].

Below we provide a brief outline for each of them, grouping them by topic where this facilitates presentation.

2.3.1 Equivalence Relations Based on ODEs

Similarly to behavioural equivalences for more classical models of computation (e.g., labelled transition systems (LTS)), the main objective of developing equivalence relations for ODEs is to support model reduction and comparison. This study has been conducted at different semantic levels, which are briefly summarised in this paragraph.

Deliverable 1.2 discusses a result of foundational nature directly established for autonomous nonlinear ODE systems [24]. There, the focus is on *heterogeneous* models, where heterogeneity is expressed in terms of classes of ODE variables having the same dynamics structurally, but which are characterised by distinct parameters[3]. The main contribution of [24] is a protocol that transforms the heterogeneous model into a homogeneous one which can be amenable to exact aggregation. Here, for convenience we summarise these notions of exact aggregation, which are also at the basis of the works reported in this document:

1. *Uniform lumpability*, a generalisation of a notion developed in [85] specifically for Markovian process algebra: two ODE variables are uniformly lumpable whenever they have equal solutions when starting from the same initial conditions.

2. *Exact ODE lumpability*, relating ODE variables whenever their aggregate dynamics (i.e., their sum), can be written in a self-consistent way as a single variable.

Let us give an illustrative example to show these equivalences at work. Consider the ODE system

$$\dot{x}_1 = -k_1 x_1 x_3 \qquad \dot{x}_2 = -k_1 x_2 x_3 \qquad \dot{x}_3 = -k_2 x_3 \tag{1}$$

where the "dot" notation indicates time derivative, and k_1, k_2 are constants. In this case, assuming that $x_1(0) = x_2(0)$ it can be seen that their derivatives at time $t = 0$ are equal. This implies that variables x_1 and x_2 are uniformly lumpable, that is, $x_1(t) = x_2(t)$ for all t. In this example, they are also exactly ODE lumpable. Indeed, computing the sum of their derivatives we get:

$$\dot{x}_1 + \dot{x}_2 = \dot{x_1 + x_2} = -k_1(x_1 + x_2)x_3$$

By the change of variable $y = x_1 + x_2$ we obtain a smaller ODE system which does not depend on x_1 and x_2:

$$\dot{y} = -k_1 y x_3 \qquad \dot{x}_3 = -k_2 x_3$$

Thus we have that $y(t) = x_1(t) + x_2(t)$ for all time points. We note that exact ODE lumpability does not require the initial conditions to be equal, unlike uniform lumpability. On the other hand, with uniform lumpability one can exactly recover the original model, whereas the trajectories of the exactly ODE lumped variables cannot be recovered in general. Finally, we remark that in general the two notions of aggregation are not comparable: the example above only shows an instance partition that happens to be both uniformly and exactly ODE lumpable.

[3] This work is similar in spirit to [19], where the problem of heterogeneity is considered in a qualitative setting in process algebra. There, the contribution is a new class of behavioural equivalences, called *dimming relations*, which makes two processes equivalent whenever they can match each other's actions provided that they are in the same given partition block.

Differential equivalences for process algebra. The theme of comparing ODEs has been lifted to a process algebra setting, significantly extending the aforementioned work of [85].

In [20] the notion of *differential bisimulation* is introduced for a slight variant of the process algebra studied in [85]. Similarly to Markovian bisimulations yielding an aggregated Markov process in the sense of the theory of lumpability [38], differential bisimulation yields a partition of the ODEs underlying a process algebra term in the sense of exact ODE lumpability (hence the sum of the ODE solutions of the same partition block is equal to the solution of a single, lumped, ODE). The underlying ODE-related dynamical property captured by this notion is different from [85]; in fact, it can be proven that the two equivalences are not comparable. Furthermore, the conditions of *label equivalence* of [85] require reasoning over an uncountable state space (the latter is basically represented by all the values that the ODE drift can take). By contrast, differential bisimulation is defined in terms of two notions that can be verified using only syntactic checks. This enables the adaptation to a continuous-state semantics of proof techniques and algorithms for finite, discrete-state, labelled transition systems. In particular a polynomially efficient partition-refinement algorithm is provided to compute the coarsest ODE aggregation of a model according to differential bisimulation.

Still in the context of process algebra, further work has been done concerning approximate aggregations, preliminarily in [25] and subsequently revisited in [23]. Here, the significant advance is also with respect to [85]. *Nearby processes*, i.e., process terms that have the same structure but may differ in their rate parameters are made symmetric after a perturbation of those parameters. It is proven that small perturbations yield nearby differential trajectories. Additionally, these papers consider an extended process algebra that unifies two synchronisation semantics that are well studied in the literature, useful for the modelling of computer systems and chemical networks, respectively. In both cases, numerical evidence shows that, in practice, many heterogeneous processes can be aggregated with negligible errors.

Forward and backward bisimulations for chemical reaction networks. Equivalence relations are studied specifically for chemical reaction networks (CRNs) with the well-known mass-action semantics in [16]. In particular, two equivalences over species of the CRN are introduced, developed in the Larsen-Skou style of probabilistic bisimulation, that are based on exact ODE lumpability and uniform lumpability, respectively. More specifically, *forward CRN bisimulation* is shown to be a sufficient condition for exact ODE lumpability while *backward CRN bisimulation* characterises uniform lumpability. To enhance the usefulness of these notions, [16] presents a *template* partition-refinement algorithm that is parametric with respect to the equivalence of interest, computing the coarsest refinement up to either variant in polynomial time. These equivalences can be used as an automatic model reduction tool. Indeed, two algorithms provide the quotient CRN induced by either bisimulation. With a prototype implementation available at `sysma.imtlucca.it/crnreducer/` and discussed in detail in Deliverable D5.2, CRN bisimulations are shown to reduce a number of case studies taken from the literature. Specifically, they yield quotient CRNs with number of reactions and species up to four orders of magnitude smaller than the original CRNs, leading to speed-ups in the ODE solution runtimes of up to five orders of magnitude. In two cases, it was possible to analyse models that were otherwise intractable due to excessive memory requirements. For instance, the ODE model in Eq. (1) is amenable to forward and backward bisimulation because it corresponds to the CRN

$$X_1 + X_3 \xrightarrow{k_1} \emptyset \qquad X_2 + X_3 \xrightarrow{k_1} \emptyset \qquad X_3 \xrightarrow{k_2} \emptyset$$

2.3.2 Combining Reduction Techniques with Model Checking

In [17], Feng and Hillston develop a technique which can be used to drastically reduce the run time of simulations of population CTMC models, such as those derived from PALOMA or CARMA models. This technique, inspired by previous work in combustion modelling [69], identifies populations within the model which can be ignored or removed without having significant impact on identified target populations. The error relationship between populations or agents within the system is built up using a directed graph. In the work of Lu and Law [69], the dynamics of the system are captured by a set of ODE and the graph is built via syntactic analysis

of these equations. However in the work of Feng and Hillston [17], the models considered have a CTMC semantics and the evolution, and therefore the potential error arising from removing an agent, changes over time. To address this challenge, Feng and Hillston build the directed graph over a number of preliminary simulation runs, which record the average rates of interactions and derive the potential error on this basis. From these rates the directed relation graph for error propagation is constructed, and used to prune agents from subsequent simulation runs up to some predefined error threshold with respect to the target populations. The experimental results show that significant speed-ups can be achieved (e.g. $> 70\%$), even with relatively low error thresholds such as 2.5%.

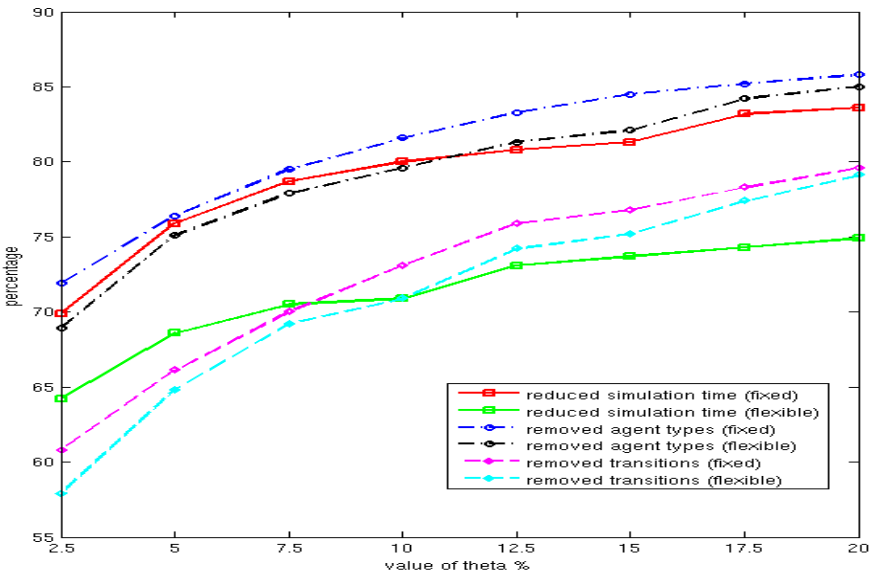

Figure 1: The percentage reduction of simulation time, agent types and transitions (y-axis) with different error thresholds (x-axis) and different variants of the reduction algorithm [17].

We anticipate that the benefit to be gained from the approach could be particularly valuable in statistical model checking where many thousands of simulation runs are usually required in order to check whether a hypothesis holds. For instance, for a model of a bike-sharing system, suppose we want to check whether the following hypothesis holds: $\Pr(\mathbf{G}_{[0,100]} 0 < x_b < C) \geq 95\%$ where x_b is the number of bike agents in a station, C is the capacity of that station. This means we require that in the first 100 time points, the probability of the station being empty or full should be less than 5%. Thus, if we set the bike agents in that station as our target agent type, the simulation speed can be significantly boosted by removing those agent types and transitions that are loosely-coupled to the target agent type, as illustrated by the sample simulation runs presented in [17] (see Fig. 1). Specifically, empirical evidence suggests that the removable agents types can be identified with a relatively small number of simulation runs, far fewer than would generally be needed to reach the criterion to accept or reject a hypothesis in statistical model checking. We plan to explore and exploit this promising application of the directed graph error propagation approach to statistical model checking in future work.

2.3.3 Automatic Moment-closure Analysis of Population CTMC models

In [18], Feng, Hillston and Galpin propose an approach for automatic moment-closure analysis of population CTMC models. The context in which the approach is developed, is PALOMA [55], but can be readily extended to any population CTMC models. The work shows how to derive a set of coupled ODEs which describe the evolution of the first-order, the second-order, and the second-order joint moments of an arbitrary population CTMC model. For CAS systems, the number of agent classes is often very large, thus the number of ODEs for the second-order joint moments can also grow very large, which may lead to a slow-down or even failure to

numerically solve the ODEs. In order to overcome this problem, a neighbourhood relation between population variables is formally defined to quantify the correlation between population variables. Then, the second-order joint moment of two population variables can be approximated by the product of their first moments if the neighbourhood relation between these two population variables is below a certain threshold. As a result, the number of ODEs can be significantly reduced. The reduced set of moment ODEs are still unresolvable as they contain moment variables that are higher than the second-order. In order to close the moment ODEs, a heuristic algorithm is first applied to reduce the highest order of moment variables to the third-order by extracting a least correlated population variable out of a moment variable that is higher than the third-order each time recursively. When the highest order of moment variables is reduced to the third-order, the lognormal moment closure [81] is applied to close the moment ODEs at the second-order. Thus, the set of ODEs can be numerically solved. Finally, the applicability, accuracy and efficiency of the moment-closure analysis approach are demonstrated by three case studies including a classic epidemic model, a wireless sensor network model and a city bike-sharing model. The moment-closure technique has been implemented in the PALOMA Eclipse plug-in that supports the recently proposed PALOMA process algebra [55]. Further details on the PALOMA Eclipse plug-in can be found in Sect. 4.4 of Deliverable 5.2 "A CAS-SCEL implementation for smart-city modelling".

2.4 Novelty and Future Work

The notions of differential bisimulation for process algebra and forward/backward bisimulation for CRNs, together with their supporting partition refinement algorithms to compute the respective largest equivalences, are entirely novel to the best of our knowledge, as is the combination of differential inequalities and exact reduction methods in order to perform approximate aggregations of ODE systems. A further innovative achievement is the model reduction technique in the moment-closure analysis of population CTMC models based on neighbourhood relation between population variables. A number of items for future work can be identified:

- Differential bisimulation and forward CRN bisimulation are sufficient conditions for aggregation. It would be interesting to weaken their assumptions in order to be able to obtain coarser aggregates. For differential bisimulation, a possible strategy would be to give up compositionality, allowing for instance two processes not be distinguished if they perform different actions that have nonetheless the same effect on the underlying ODE system.

- The effectiveness of the CRN bisimulations should be also evaluated on further model repositories than those considered in [16], for instance by interfacing with the well-known SBML language (`sbml.org`). CRN bisimulations work with the assumption of mass-action semantics. It would be interesting to extend these to other popular kinetic mechanisms in systems biology, for instance Hill and Michaelis-Menten kinetics. Even if the QUANTICOL project is not specifically focussing on applications in system biology, some of them can be seen as instances of CAS and the results obtained appear to have interesting applications in that area which are worth pursuing further. Furthermore, systems biology provides one with a rich repository of realistic large-scale models (see also Deliverable 5.2) which can be used to study the effectiveness and scalability of the proposed reduction techniques.

- Differential hulls provide satisfactory bounds for small enough degrees of heterogeneity. We will investigate other techniques to obtain tighter bounds. The developments on uncertain ODE models developed in WP1 and reported in Deliverable D1.2 appear to be promising in this respect.

- Investigation of classical abstractions at the level of CTMC, with the aim of linking local and global properties of systems. We plan to focus mainly on approximate lumpability relations generating time-inhomogeneous lumped Markov Chains and on discretisations of the mean field limit, generating small Markov Chains capturing the relevant stochastic behaviours.

- Combining reduction techniques with Fluid Model Checking. Several reduction techniques mentioned in this section could be combined with model-checking techniques such as statistical model checking but

also fluid and mean field model checking. It would be interesting to explore several of these possibilities in the context of CAS.

3 Relating Local and Global System Views with Variability Analysis

3.1 State of the art/baseline before start of the project

Software Product Line Engineering (SPLE) [43, 74] is a software engineering approach that aims to develop and maintain a family of software-intensive systems by the systematic reuse of components or subsystems from related products or earlier versions of a product. To do so in a cost-effective manner, the engineering process is organised so as to maximise commonalities and at the same time minimise the cost of variations among individual products. Therefore individual products share an overall (global, collective) reference model or architecture of the product family, but they differ with respect to specific (local) features. SPLE reduces time-to-market, increases product quality and lowers production costs.

A feature diagram or *feature model* is the de facto standard variability model in SPLE [79]. A *feature* characterises a stakeholder visible piece of functionality of a product or system and a feature model provides a compact representation of all possible products of a product line or configurable system in terms of their features (behaviour is not captured). However, there may be hundreds of features or configurable options, which easily leads to superfluous or contradictory variability information (e.g., 'false' optional or 'dead' features). There is a lot of work on computer-aided analyses of variability models to extract valid products and to detect anomalies [36]. Graphically, features are nodes of a rooted tree and relations between them regulate their presence in products (e.g., a *requires* constraint indicates that the presence of one feature requires that of another, cf. Fig. 2). A product $\mathscr{P}$ from the SPL is identified by a non-empty subset $\mathscr{P}_{\mathscr{F}}$ of the set $\mathscr{F}$ of features. Deciding whether a product satisfies a feature model can be reduced to Boolean satisfiability (SAT), which can be effectively computed with SAT solvers [31].

The common (global) and variable (local) parts of products are thus defined in terms of features, and managing variability is about identifying the variation in a shared family model to encode exactly which combinations of features constitute valid products. The actual configuration of products during application engineering is then reduced to selecting desired options in the variability model. One of the aims of Task 3.3 is to study the relationship between the micro and macro views of a CAS. Inspired by variability analysis as known from SPLE, the idea is to define the macro view by indicating the commonalities and variability among a collection of micro views, and to adapt established variability analysis techniques to obtain results concerning the kind of properties that can be preserved from macro to micro view and vice versa.

The state of the art in variability modelling and analysis of software-intensive systems focussed on structural rather than behavioural properties and constraints. It is of course important to model and analyse variability also at the behavioural level, in order to provide a form of quality assurance. Before the beginning of QUANTICOL, this was starting to gain popularity. A lot of research in SPLE was concerned with lifting successful modelling languages and formal verification techniques known from single (software) system engineering, such as process calculi — and other specification languages with a semantics in terms of Labelled Transition Systems (LTSs) — and model checking, to SPLE [83]. The challenge is to handle the variability inherent to Software Product Lines (SPLs), by which the number of possible products of an SPL may be exponential in the number of features. Therefore, in variability analysis, one distinguishes *product-based* analyses, operating on individually generated products or at most a subset, from *family-based* analyses, operating on an entire SPL at once using variability knowledge about valid feature configurations to deduce results for products [83]. In family-based verification, once a property is verified for a family model one knows that the result also holds for any of its product models, without the need to explicitly verify the property over the products; this is in general more efficient than product-based verification, in which every product thus has to be examined individually.

The most widely studied semantic models to capture in a compact way all possible operational behaviour of the products of a product family were based on *Featured Transition Systems* (FTSs) and on *Modal Transition Systems* (MTSs).

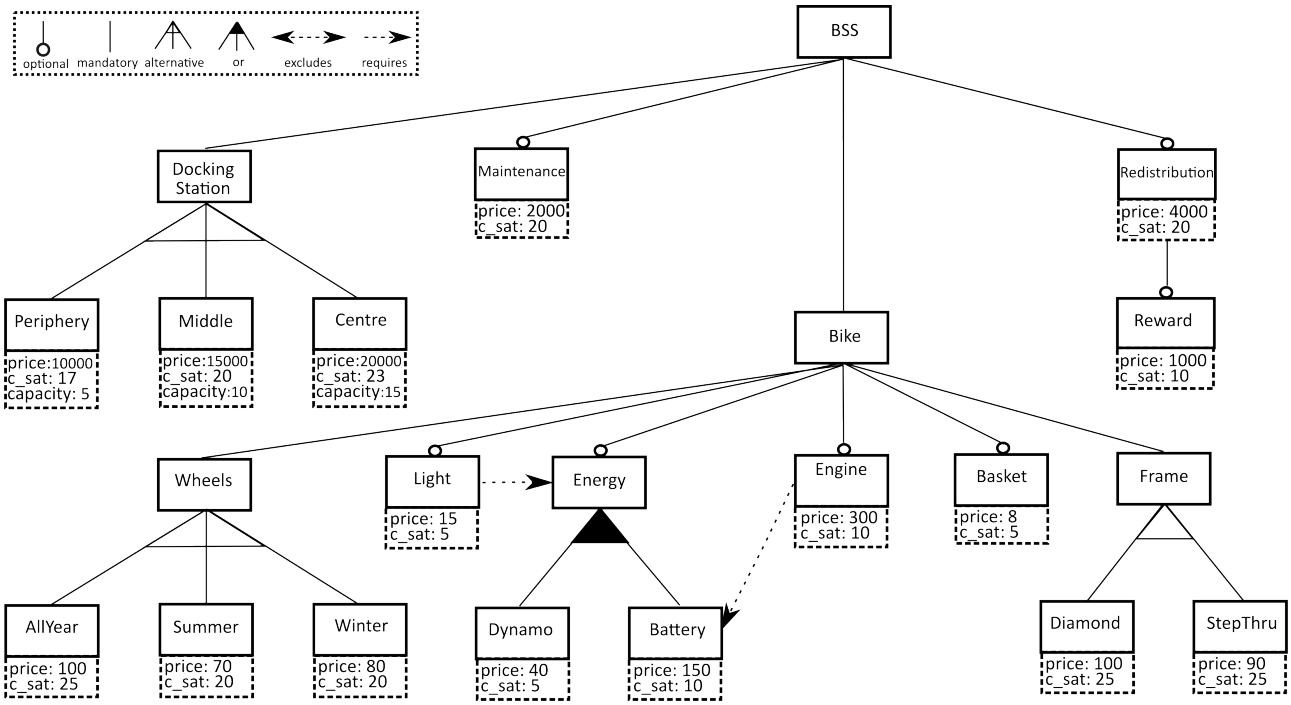

Figure 2: Attributed feature model of a Bike-Sharing System BSS.

An FTS [42] is an LTS equipped with an additional feature diagram. Each transition of the LTS is labelled by an action and, using the improved definition from [40], an associated feature expression (a Boolean formula defined over the set of features) that needs to hold for this specific transition to be part of the executable product behaviour. Hence an FTS models a family of LTSs, one per product, which can be obtained by projection: all transitions whose feature expression is not satisfied by the specific product's set of features are removed, as well as all states and transitions that because of this become unreachable. An MTS [65, 26] is an LTS that distinguishes between admissible ('may') and necessary ('must') transitions. Considering the may transitions as optional and the must transitions as mandatory, an MTS can be interpreted as a family of LTSs such that each family member corresponds to a specific selection of optional transitions. In this way, a single MTS can model a product family since it allows a compact representation of the family's behaviour, by means of states and actions, shared between all products, and variation points, by means of may and must transitions, used to differentiate among products. However, it is well-known that plain MTSs cannot capture all variability notions known from feature models (e.g., 'alternative' and 'mutually exclusive' features) without adding additional constraints.

Finally, once one equips features with attributes (e.g., $\texttt{capacity}(\texttt{Centre}) = 15$ in Fig. 2) one obtains an *attributed feature model*. Think, for example, of feature attributes related to non-functional aspects like price, weight, reliability, etc. Now a product $\mathscr{P}$ from the SPL is a non-empty subset $\mathscr{P}_{\mathscr{F}} \subseteq \mathscr{F}$ that moreover satisfies the additional *quantitative constraints* over feature attributes (e.g., $\texttt{capacity}(\texttt{DockingStation}) \leq 10$). Complex quantitative constraints require the use of computationally more expensive Satisfiability Modulo Theories (SMT) solvers like Microsoft's Z3 [71]. While specifically modelling and analysing variability also at the behavioural level was gaining popularity at the beginning of QUANTICOL, quantitative constraints over feature attributes were hardly taken into account, while probabilistic SPL models or performance analysis of SPL models were not yet considered. Probabilistic modelling and performance analysis are important, though, since they allow one to model uncertainty, failure rates and randomisation, thus permitting (quantitative) analyses measuring quality of service, reliability or performance.

3.2 Objectives

Task 3.3 has a number of objectives, which can be summarised as follows.

1. To obtain a better understanding of the relation (e.g., in terms of the preservation of properties) between local (micro) and global (macro) views of a system or CAS by using the analogy with common (global) and variable (local) features of a family of products.

2. To develop process-algebraic SPL modelling languages with a semantics that specifically allows for *family-based* variability analysis (e.g., by means of model checking) and performance evaluation.

3. To explicitly consider *quantitative* feature constraints and *probabilistic* behaviour in the specification and analysis of SPLs and CAS seen as such (e.g., in the context of bike-sharing systems).

4. To lift behavioural equivalence relations from LTSs to SPL models (e.g., FTSs) to support *model reduction* and comparison, as a step towards scalable model checking of behavioural SPL models.

3.3 Achievements in the second reporting period

The achievements obtained during the reference period addressed in this deliverable concerning the first phase of Task 3.3, from month 7 upto month 30, are described in detail in the following publications: [5], [22], [3], [21], [14], [4], [11], [12], [2], [9], [1], [10], [13], [6], [7], [8], [15]. The following sections briefly describe these results.

Part of this work was originally planned in phase 2 of T3.3, but has been addressed a little earlier.

3.3.1 Variability from a Macro and Micro View Perspective

In [12], the full formal underpinnings of a modelling and analysis framework for the specification and verification of variability in product families is presented. Some of its aspects were introduced before, but several new notions are introduced, like *consistent* and *valid* product LTSs. Variability is addressed at the behavioural level by modelling the family behaviour by means of a specific subset of MTSs, which are moreover equipped with an additional set of logical variability constraints expressed over actions. These constraints allow the capture of all common variability notions known from feature diagrams (which plain MTSs cannot). Steered by the variability constraints expressed over action labels, the inclusion or exclusion of labelled transitions in an LTS refining the MTS determines the family's possible product behaviour (modelled as LTSs). This is formalised as a special-purpose refinement relation for MTSs, which differs fundamentally from the classical one [26], and it is shown how to use it for the definition and derivation of valid product behaviour starting from product family behaviour. This special-purpose refinement relation always preserves the exact original branching structure (upto the removal of unreachable states) of the MTS in the product LTSs, thus cutting out LTSs with unfoldings and duplications (and with unreachable states) that are responsible for the fact that the number of LTSs obtained by classical refinement is in general infinite. Consistency concerns the fact that a feature (i.e., functionality) is or is not present in a product, independently of its behavioural context. This means that when an optional transition with a specific label is included in a product LTS, this must be done in a consistent way for all its occurrences. A consistent product is valid if it moreover satisfies all variability constraints.

To reason about properties of MTS models, v-ACTL (a *variability-aware action-based branching-time modal temporal logic* interpreted over MTSs, cf. [5]) is used. A number of results regarding the preservation of logical properties from family (macro) to product (micro) behaviour is formally defined and proved in [12]. These results pave the way for the more efficient (global) family-based analyses of MTSs, limiting the need for costly (local) product-by-product analyses of LTSs. Finally, [12] contains the full syntax and semantics of the high-level modal process algebra used for the specification of MTSs, which is moreover extended with *value-passing* communication in [4]. Fig. 3 depicts the result of verifying the v-ACTL formula

$$AG\,EF^{\Box}\,\{\texttt{givebike(s1)}\}\,true \tag{2}$$

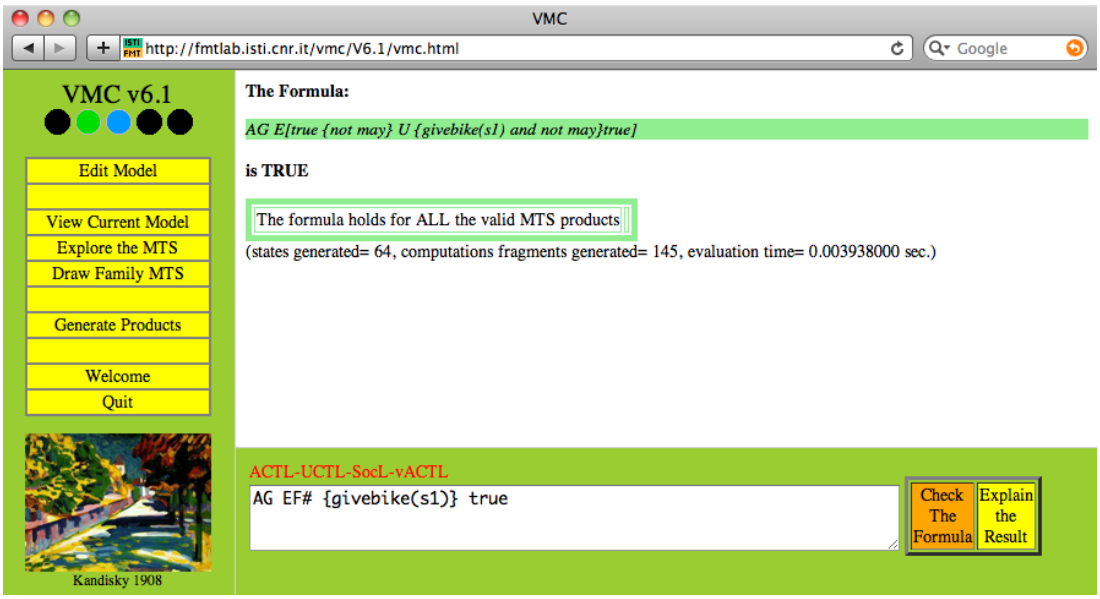

Figure 3: Result of a family-based analysis with the VMC model checker.

using VMC (described below) over a family of bike-sharing systems specified in the value-passing modal process algebra accepted by VMC. Formula (2) expresses the property that for all products it is always the case that there exists a path in which eventually docking station 1 ($s1$ in the formula) must give a bike to a user, thus providing the requested service. The operator $F^{\Box}$ is specific to v-ACTL as it takes the modality of the transitions (i.e., *may* or *must*) into account.[4] The fact that this property requires a 'must path' is the reason that this is a property that holds for all valid products (i.e., LTSs) of the MTS whenever it holds for the family, as VMC reports.

VMC: an on-the-fly variability model checker. The complete framework described above was implemented in a model-checking tool developed before the start of the project: the *Variability Model Checker* VMC [34] (`fmt.isti.cnr.it/vmc`). Given the behaviour of a product family modelled as an MTS with an additional set of variability constraints, it allows the explicit generation of valid product behaviour and the resulting LTSs can be verified against a logic property expressed in v-ACTL. VMC was extended during the project in [5] such that a logic property expressed in v-ACTL can be verified directly over the MTS, relying on the fact that under certain syntactic conditions validity over the MTS guarantees validity of the same property for all the family's valid products (LTSs). This provides a relation between global macro views and local micro views.

VMC is the most recent product of the KandISTI family of model checkers [32] (`fmt.isti.cnr.it/kandisti`) developed at ISTI–CNR over the past two decades, which includes also FMC [58], UMC [35] and CMC [54]. Each of them allows for the efficient verification, by means of explicit-state on-the-fly model checking, of functional properties expressed in the *KandISTI Temporal Logic* (KTL), whose full syntax has been developed during the project and is provided in [2]. KTL is an action-based and state-based branching-time temporal logic derived from the family of logics based on ACTL [49, 51], i.e., action-based CTL, and encompasses v-ACTL.

The model checkers of KandISTI share an optimised verification engine, exploiting an on-the-fly analysis approach such that only part of the state space needs to be generated and explored. Their input models are rather different though, due to the varying fields of application for which they were developed: VMC has been discussed above; FMC accepts automata networks integrating the communication and synchronisation mechanisms from the classical process algebras CCS, CSP and LOTOS [56] in a single process-algebra, thus allowing

[4] The intuitive interpretation of $F^{\Box}\{\psi\}\,\phi$ is that there exists a future state, reached by an action satisfying ψ, in which ϕ holds, and all transitions on the path until that state are *must* transitions. In VMC, $F^{\Box}$ must be written as `F#`.

both multi-way synchronisation and value-passing; UMC accepts systems specified as sets of communicating UML-like state machines; CMC, finally, accepts systems specified in a calculus for the orchestration of web services.

In [11], it was shown how also the FMC model checker (`fmt.isti.cnr.it/fmc`) and its KTL logic can be quite easily used for the modelling and (family-based) analysis of behavioural variability in SPLs. FMC is a modelling and verification framework for the definition, exploration, analysis and model checking of system designs modelled as parallel compositions of sequential terms. FMC accepts parameterised specifications and it supports the verification of properties expressed in a logic that specifically allows one to correlate the parameters of different actions within a formula. This feature, together with FMC's lazy evaluation and special-purpose predicates, can be used to tailor formulas to the verification of a specific subset of products of an SPL, thus allowing for family-based analyses of SPLs. The complexity of verifying a formula for a subfamily (i.e., a subset of products characterised by a subset of features) obviously does not depend on the size of the rest of the family. In combination with the on-the-fly model-checking algorithm of FMC, this means that more often than not only part of the complete state space needs to be inspected, which considerably improves the scalability of our approach.

FTS vs. MTS. The relation between FTSs and MTSs as used in SPLE still needs to be studied in full detail, both with respect to their semantic models and their associated model checkers. FTSs come with dedicated SPL model checkers like SNIP [41], which is now integrated and reengineered in the product line of model checkers ProVeLines [44], allowing efficient family-based analyses capable of relating errors and undesired behaviour to the exact sets of products in which they occur. In [9], an automatic technique is provided to transform FTSs into MTSs with additional sets of variability constraints in the specific format accepted by VMC. The crux of this transformation is thus to go from variability constraints expressed in terms of features to variability constraints expressed in terms of actions. Formally, an FTS is a doubly-labelled LTS, in which each state is labelled by an atomic proposition, but this labelling is ignored in [9]. In the future, we intend to consider also the state labelling of FTSs by switching from a purely process-algebraic description of MTSs in VMC to a richer modelling language. Other KandISTI members in fact have both an action and a state labelling.

While the behaviour of an SPL can, of course, be directly specified in FMC, [11] describes a technique to automatically transform an FTS into a process-algebraic model in the specific format accepted by FMC, thus paving the way for a comparison of the modelling and analysis of SPL behaviour also in this model-checking framework.

To conclude, we studied two transformations from FTS into the MTS framework (both illustrated on the same bike-sharing example). These transformations serve two purposes. First, they contribute to a better understanding of the fundamental differences between the two approaches (i.e., expressing variability constraints over actions with or without an associated feature label). Second, they pave the way to compare the modelling and analysis of SPL behaviour in three different settings. However, a detailed evaluation and comparison is left for future work.

Behavioural relations for product lines. An alternative approach to formally reasoning about SPL is presented in [22], which focusses on behavioural relations, such as algebraic simulation and bisimulation, instead of verification through model checking. This is done in the context of *Variant Process Algebra* (VPA), a calculus endowed with an operational semantics mapped onto LTSs, and designed along the lines of CCS and CSP. The syntax, however, is minimally extended in order to promote *variants* to become first-class citizens of the language. In essence, the underlying idea in VPA is to tag a process term with the set of variants where it is enabled. The syntax of the (binary) operator for parallel composition is extended in VPA in a similar fashion, by using two sets of variants that specify where the synchronising processes are enabled. In more detail, VPA has the following features:

1. A multi-modal semantics, where *each variant* gives rise to a distinct transition relation over VPA. The approach we take is different from the literature on extensions of MTSs in that variants are explicitly

enumerated. The approaches based on MTSs, instead, distinguish between mandatory (*must*) and optional (*may*) behaviour through two different transition relations [57, 64, 53, 68, 28]. In this respect, VPA is more closely related to [29] whereby, while still considering *may* and *must* transition relations, a semantics is given to explicitly derive all the products of a family.

2. VPA has a *family-based semantics* which leads to an LTS that characterises the behaviour of the whole family. For every VPA process, the behaviour of any variant is *simulated* by the behaviour induced by the family-based semantics. In this sense the use of simulation as a useful means of relating a single product with the whole product line (e.g., [53, 42, 29]) is lifted to process algebra.

3. Finally, the ability to explicitly label variants allows reasoning about the relationship between them. VPA introduces the notion of *variant simulation* which can be useful to establish that a variant v_1 can be regarded as a conservative extension of another variant v_2 in that it contains all of v_2's behaviour, and possibly more. In some cases it is possible to find a variant that can simulate every other variant (or possibly large subsets thereof). This may bring about the advantage that one can study the behaviour of such a variant instead of the model induced by the family-based semantics, which may turn out to be more complex (e.g., because of more states and/or transitions). Sufficient conditions, solely based on syntactic checks on a process term, are provided in order to establish variant simulation for VPA.

3.3.2 Quantitative Variability Analysis and Applications to Bike-Sharing Systems

We have obtained some initial results concerning extensions of the current qualitative approaches for the verification of SPLs to quantitative and scalable approaches that can be used in the context of CAS. This research has benefitted from our ongoing collaboration with PisaMo S.p.A. (`www.pisamo.it`), an in-house public mobility company of Pisa's administration that has introduced the public bike-sharing system *CicloPi* in Pisa two years ago, and its supplier Bicincittà S.r.l. (`www.bicincitta.com`). They have generously shared with us their knowledge on bike-sharing systems in general and on *CicloPi* in particular. This bike-sharing system currently has only 15 stations and roughly 140 bikes, which makes it an ideal case study to start with.

In [3], SPLE techniques were used to define a family of bike-sharing systems, after which support for variability analysis was sought among available tools. This resulted in the *tool chain* depicted in Fig. 4, including the (academic) tools S.P.L.O.T. [70], FeatureIDE [84], Clafer [30], ClaferMOO [72] and VMC [34, 5]. It provides different functionalities regarding the analysis of SPLs, from feature modelling to product derivation and from quantitative evaluation of the attributes of products to model checking value-passing modal specifications. The tool chain served as a first experiment towards more sophisticated variability analysis techniques.

To address also quantitative analysis, in [2] the SPLE paradigm was considered at the level of system engineering by first defining a reference feature model and then adding feature attributes and global quantitative constraints, in the form of an attributed feature model in the Clafer toolset. In particular, its ClaferMOO(Visualizer) allows one to compare system configurations (variants) with respect to various quality dimensions (e.g., cost), select the most desirable one and analyse the impact of reconfigurations on a variant's quality dimensions. In fact, it was used for quantitative analyses and multi-objective optimisation of the resulting attributed feature model.

A preliminary study towards automatic *decision support* for the initial design of a bike-sharing system, as well as its successive adaptations and reconfigurations, taking both qualitative and performance aspects into consideration, is described in [14]. To this aim, two complementary strategies for the evaluation of bike-sharing system designs by means of automated tool support are presented. The Clafer toolset was used to perform multi-objective optimisation of variability models with feature attributes and cardinalities (in particular the one depicted in Fig. 2 and their multiplicity in a particular BSS configuration) while the recently developed mean field model checker FlyFast [66, 67] was used to assess performance and user satisfaction aspects of variants of large-scale bike-sharing systems. In particular, the bike sharing model studied in the first reporting period (see Deliverable 3.1) has been extended to model a heterogenous system with stations with different capacities and bike request and return rates that vary across the areas of the city. It has also been shown how to obtain commonly used indicators such as the normalised bike availability per station (NAB) from the data

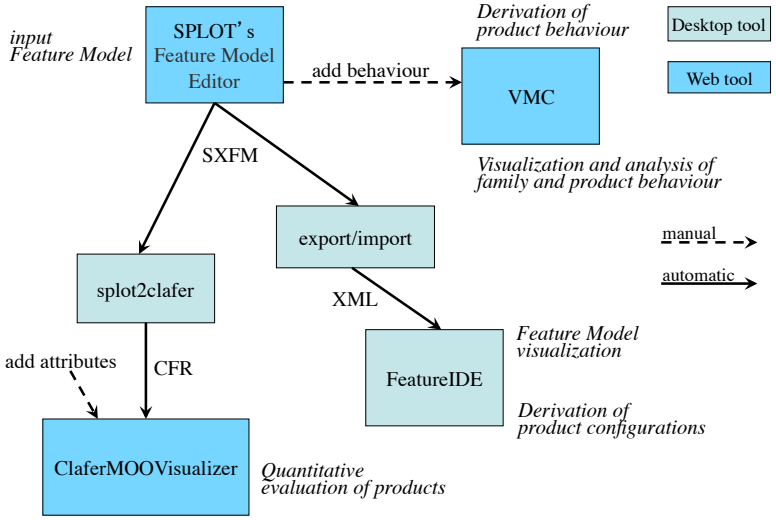

Figure 4: Tool chain used for the experiments described in [3].

generated by the mean field models which facilitates comparison between models and real data. The focus was on a simplistic comparison of two different BSS configurations with respect to their cost, user satisfaction and capacity (in terms of parking slots) to illustrate the ideas. A future goal could be to strengthen the integration of these two approaches and use the outcome of performance analyses as input for variability modelling. One could, for example, measure the user satisfaction for specific configurations and feed the resulting values into a variability model. Performance analyses such as those performed in [14] (e.g., *the probability of finding an empty/full docking station given the specific capacity of a BSS configuration*) clearly have an impact on user satisfaction. Moreover, the approach of [14] was showed to easily scale to bike-sharing systems of realistic size.

Finally, [1] explores the possibility of applying a *machine learning* approach to implement features and, at the same time, evaluate them to derive meaningful values to fill the (attributed) feature model. It concentrates on predictive features that are able to analyse the current state and some historical data, and provide some information to the user. More generally, the purpose of the analysis is to evaluate the features and their possible combinations to help a stakeholder to decide which product of a line to deploy, making the best possible compromise between cost and usefulness.

FLan: a family of feature-oriented languages. The FLan family of *Feature-oriented Languages* (FLan [33], PFLan [10] and QFLan [13]) contributes to ongoing efforts to lift successful specification languages and verification techniques from single system engineering to SPLE. The FLan family is inspired by the concurrent constraint programming paradigm of [75], its adoption in process calculi [39] and its stochastic extension [37]. In FLan, a rich set of process-algebraic operators allows one to specify both the configuration and the behaviour of products, while a constraint store allows one to specify all common constraints known from feature models as well as additional action constraints reminiscent of FTSs. The execution of a process is constrained by the store (e.g., to avoid introducing inconsistencies), but a process can also query the store (e.g., to resolve configuration options) or update the store (e.g., to add new features, even at run time). Its implementation in Maude allows analyses ranging from consistency checking (by means of SAT solving) to model checking.

This line of research was continued by investigating the suitability of statistical model-checking techniques for the analysis of probabilistic models of SPLs with complex quantitative constraints and advanced feature

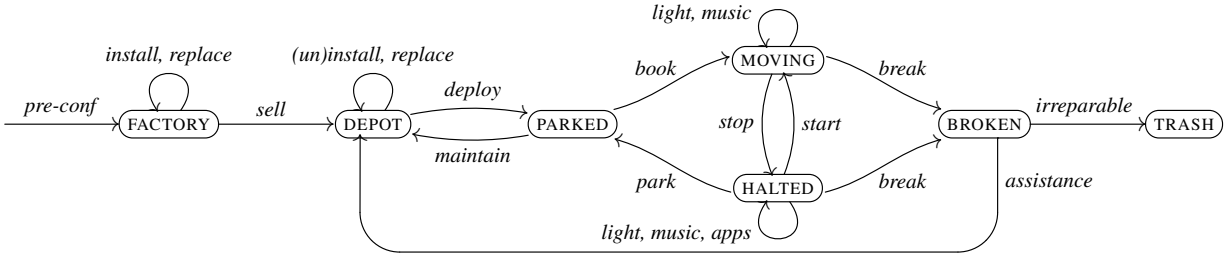

Figure 5: Sketch of bike-sharing behaviour, omitting the action probability weights.

installation options. During the project, FLan was equipped with the means to specify *probabilistic* models of SPLs, resulting in PFLan [10]. The main distinguishing modelling feature of FLan is the clean separation between an SPL's configuration and run-time aspects. PFLan adds to this the possibility to equip each action (including those that install a feature, possibly at run time) with a probability weight, which can represent uncertainty, a failure rate, randomisation or simply relative preferences, resulting in a DTMC model after appropriate normalisation of the weights. An efficiently executable implementation in Maude, together with the distributed statistical model checker MultiVeStA [80], developed during the first reporting period of the project and described in Deliverable 5.1, allows one to estimate the likelihood of specific configurations and behaviour of an SPL, and thus to measure non-functional aspects such as quality of service, reliability or performance.

QFLan: probabilistic SPL models with quantitative constraints. QFLan, which was introduced in [13], further extends PFLan with dynamic uninstallation and replacement of features and with advanced *quantitative constraint modelling* options. This allows for more involved quantitative analyses, now requiring SMT solving. This was achieved by integrating an efficiently executable Maude implementation of QFLan with Microsoft's Z3 [71] and with MultiVeStA.

The advanced quantitative constraint modelling options concern the 'cost' of features, i.e., feature attributes related to non-functional aspects such as price, weight, reliability, etc. In particular, the novel modelling options introduced in [13] are:

1. Arithmetic relations among feature attributes (e.g., the total cost of a set of features must be less than a certain threshold);

2. Propositions relating the absence or presence of a feature to a quantitative constraint as in 1 (e.g., if a certain feature is present, then the total cost of a set of features must be less than a certain threshold);

3. Richer action constraints involving quantitative constraints as in 1 (e.g., a certain action can be performed only if the total cost of the set of features constituting the product is less than a certain threshold).

The uninstallation and replacement of features can be the result of malfunctioning or of the need to install a better version of the feature (e.g., a software update). It is important to note that the above type of quantitative constraints are significantly more complex than the ones that are commonly associated to attributed feature models. Moreover, quantitative constraints based on arithmetic relations between feature attributes are a novel contribution of QFLan.

Quantitative variability analyses of a bike-sharing scenario. The resulting modelling and analysis framework was applied to a bikes product line case study: essentially the `Bike` feature subtree of the attributed feature model depicted in Fig. 2 (with an additional computational unit feature) and the behaviour sketched in Fig. 5 (the action probability weigths are omitted from the drawing to avoid cluttering).

Some typical properties of interest for the bike-sharing case study are:

(P_1) Average price, weight or load of a bike either when it is first deployed or over time;

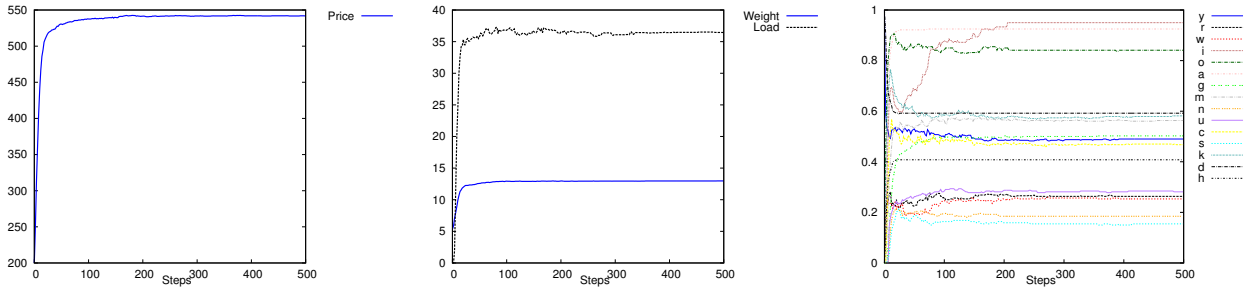

Figure 6: MultiVeStA results for P_1 and P_2: price (l), weight and load (m), installation probability (r).

(P_2) For each of the 15 primitive features (i.e., the leaves in Fig. 2), the probability to have it installed either when a bike is first deployed or over time.

Analysing P_1 and P_2 at a bike's first deployment can be useful for studying an *initial scenario*, to estimate the required initial investments and infrastructure. Bikes with a high price and load (i.e., with a high technological footprint) or equipped with a battery might, for example, require docking stations with specific characteristics or have to be collected at night for safe storage. Instead, analysing P_1 and P_2 over time provides an indication of how those values evolve, for example, to estimate the average value in euros of a deployed bike and the financial consequences of its loss. In general, properties like P_2 measure how often (on average) a feature is actually installed in a product, which is important information for those responsible for the production or programming of a specific feature or software module. Some results of analysing P_1 and P_2 over time with MultiVeStA [80] are shown in Fig. 6 for prices (left), for weights and loads (middle) and for the probabilities of installing features (right).

Fig. 6(left) shows that the average price (on the y-axis) of the intermediate bikes generated from the SPL starts at €200, in line with the initial configuration (features `AllYear` and `Diamond` installed). Then the price grows with respect to the number of performed simulation steps, consistent with the pre-configuration phase FACTORY during which a number of features can be installed, followed by a customisation phase DEPOT, where features can be (un)installed and replaced. The probability of a bike to return to the DEPOT after its first deployment is quite low (in fact, in the QFLan specification PARKED has a transition with rate 10 towards MOVING and one with rate 1 towards DEPOSIT, interpreting the probability weights as rates in this case, leading to the embedded DTMC as a model). Thus, on average, the price of bikes is only slightly affected by (un)installations and replacements performed by successive DEPOT phases. Finally, as confirmed by Fig. 6(right), the probabilities (on the y-axis) for each feature to be installed evolve similarly to the average price, weight and load of the generated products. Notably, the pre-installed feature `AllYear` (y), relating to the kind of tires, has probability 1 of being installed at step 0, after which the probability decreases during the initial steps.

Performance analysis of variant-rich software systems. The focus of [21] was to find effective means to handle variability in software performance models. In particular, we considered a class of UML Activity Diagrams (ADs), to model systems which can be reasonably described as workflow processes, such as real-world data-centers [82], service-oriented architectures [63] and automation systems [86]. To capture performance properties, ADs are augmented with annotations (such as the duration to execute an activity of an activity node) and interpreted as continuous-time Markov chains, along the lines of established routes in model-driven software performance engineering. These *performance-annotated ADs* (PAADs) are integrated with SPL techniques to precisely capture variability aspects. More specifically, we considered a *delta-oriented* approach, where possibly many variants can be generated as a result of applying changes (i.e., *deltas*) to a *core* PAAD [76]. This represented a novel application of delta modelling, which has so far been used to represent static variability of software architectures [61] and Java programs [77].

A straightforward solution technique requires a separate analysis of each variant—the so-called *product-based* evaluation. In [21], a *family-based* approach was considered. The configurable parameters of the model under study, inferred by the kinds of delta operations defined on the AD, are used for obtaining a solution in

symbolic form. In this way, performance indices can be simply obtained by evaluating a polynomial expression that explicitly depends on the configurable parameters. The evaluation may become faster than the product-based analysis, which is based on the numerical inversion of a matrix of size equal to the number of nodes in the PAAD. By numerical experimentation we show that our family-based approach is up to three orders of magnitude faster, with a tendency to become increasingly more convenient as the model size grows. Although family-based SPL analyses have been introduced for type checking [27, 52, 46] and model checking [42, 68, 28, 29], for the first time this approach is considered for the efficient performance modelling of SPLs.

3.3.3 Behavioural Model Reduction by Abstraction

For the efficient and scalable model checking of SPLs, behavioural equivalence relations to support model reduction and comparison need to be lifted from LTSs to SPL models. We present a new approach addressing this aspect.

Feature-oriented modular verification. A different approach to deal with an SPL's possibly exponential number of products is to make an existing off-the-shelf model checker amenable to model checking SPLs. We worked towards a *feature-oriented modular verification* approach, using an interpretation of FTSs in the mCRL2 language and toolset. mCRL2 is a formal specification language for the modelling of distributed systems and their interactions, which comes with a state-of-the-art toolset for the qualitative behavioural analysis [45, 59]. It has already shown its merits in dealing with huge state spaces consisting of billions of states. The mCRL2 data language allows one to smoothly deal with feature sets and attributes, its process language is sufficiently rich to model feature selection and product behaviour based on an FTS semantics, and the variant of the μ-calculus that acts as property language for the toolset supports the use of data in formulas.

In [6, 7], it was shown how mCRL2 can indeed be exploited for the modelling and analysis of SPLs. In particular, a proof-of-concept was elaborated, illustrating how to use mCRL2's parametrised data language to model and select valid product configurations, in the presence of feature attributes and quantitative constraints, and to model and check the behaviour of valid products. In [8], it was subsequently shown how the resulting behavioural model can be modularised (based on feature-driven borders) into separate components, using branching bisimulation techniques to isolate them; these components are equipped with interfaces (in the form of exit and (re-)entry transitions) that allow an additional driver process to glue them back together on the fly, exhibiting the same behaviour as before. This is a powerful abstraction technique that allows mCRL2 to concentrate on the relevant components (features) for the specific property under scrutiny, abstracting from the other components. This approach thus differs from modular or compositional verification in the classic sense of (re)composing smaller verification results on modules or components to derive properties of the composed system.

A behavioural equivalence for FTSs and model minimisation. In [15], finally, a behavioural equivalence for FTSs was proposed as a generalisation of branching bisimulation for LTSs. It was proved that *branching feature bisimulation* for an FTS of a family of products coincides with branching bisimulation for the LTS projection of each of the individual products. For a restricted notion of *coherent branching feature bisimulation*, a minimisation algorithm was presented and its correctness proved. This thus complements and formalises part of the approach outlined in [6, 7, 8]. Although the minimisation problem for coherent branching feature bisimulation is shown to be NP-complete in general, application of the algorithm in the setting of a small case study results in a significant speed-up of family-based verification by model checking behavioural properties. It remains to establish the subset of the modal μ-calculus preserved by branching feature bisimulation and apply the approach to an industrial-size SPL. The conjecture is that, exactly as for branching bisimulation, it preserves the modal μ-formulae without the next operator [50]. It would also be interesting to see whether the minimisation algorithm's complexity can be reduced, possibly by lifting some optimisations from the Groote and Vaandrager algorithm for LTS to the FTS setting [60].

3.4 Novelty and Future Work

The results presented in this deliverable contain several novel contributions:

- Family-based analysis of SPLs with a delta-oriented approach to performance-annotated UML activity diagrams is a novel application of delta modelling and the first attempt to efficient performance modelling of SPLs.
- Quantitative constraints based on arithmetic relations between feature attributes are an innovative aspect of QFLan. Moreover, the implementation of QFLan led to the first application of statistical model checking in SPLE.
- The explicit enumeration of variants in VPA, allowing one to relate concrete variants to the whole product line and variants to each other, is different from other calculi defined in SPLE.
- The transformations from FTSs into the input models accepted by the FMC and VMC model checkers are new.
- The special-purpose refinement relation for MTS models of SPLs that defines their valid product behaviour (modelled as LTSs) is fundamentally different from the classical one for MTSs.
- The feature-oriented modular verification approach developed for the mCRL2 toolset differs from modular or compositional verification in the classic sense. Moreover, the branching feature bisimulation for FTSs (and its minimisation algorithm) that is at the heart of this approach, is the first example of lifting branching bisimulation to SPL models.

A number of items for future work can be identified:

- Consider further behavioural relations for VPA (e.g., a notion of variant bisimulation) and provide logical characterisations of all such relations to study model-checking properties of VPA models.
- Equip the QFLan implementation with optimisation capabilities, so that configuration options optimising objective functions can be obtained automatically.
- Provide an FTS and/or an MTS semantics for members of the FLan family to be able to use dedicated SPL model checkers.
- Establish the subset of the modal μ-calculus that is preserved by (coherent) branching feature bisimulation (i.e., what properties are preserved by model reduction).
- Perform a quantitative evaluation of the expressivity, complexity and scalability of the modelling and verification frameworks for (family-based) behavioural variability analysis of SPLs discussed in this deliverable (i.e., those based on MTSs/FMC/VMC, FTSs/SNIP/ProVeLines, QFLan/Maude/Z3/MultiVeStA and FTSs/mCRL2).

4 Conclusions and Roadmap

In this deliverable we reported on the progress made concerning the objectives of Task 3.2 and Task 3.3 of Work Package 3 during the second half of the first reporting period and during the second reporting period of the project. In particular we addressed the development of behavioural equivalence relations for ODE to support model reduction and comparison of models and alternative model reduction techniques and the extension of techniques to deal with variability analysis in the context of collective adaptive systems.

Main achievements. Below we briefly summarise the main achievements mentioned in the individual sections:

For what concerns model reduction techniques we mention:

- The development of two new bisimulation relations for chemical reaction networks: forward CRN bisimulation and backward CRN bisimulation based on exact ODE lumpability and uniform lumpability, respectively. These yield reductions upto four orders of magnitude compared to the size of the original CRN's which in some cases made it possible to analyse models that otherwise would be intractable [16].
- The development of a model reduction technique based on the identification of populations within the model that can be removed or ignored [17].

For what concerns variability analysis for CAS we mention:

- The development of a method to effectively handle variability in software performance models combining a delta-oriented approach with performance-annotated UML activity diagrams obtaining solutions in symbolic form suitable for family based analysis [21].
- The introduction and implementation of the probabilistic and feature-oriented language QFLan which offers advanced quantitative constraint modelling options [13].
- The development of a branching feature bisimulation relation for FTSs, together with an algorithm for model minimisation [15].
- The formalisation of a special-purpose refinement relation for MTSs to determine valid product behaviour modelled as LTSs, together with the precise set of logical properties that it preserves, and the implementation in the VMC model checker for variability analysis [12].

Relationship to other work packages.

WP1 Emergent Behaviour and Adaptivity. The work on approximate aggregation for nonlinear ODEs is of direct relevance both for Task 3.2 of WP3, where the main focus is on model reduction, and for Task 1.1 of WP1, described in Sect. 3.2 of Deliverable 1.2 where the theoretical results are used to develop practical numerical methods to analyse a class of limit models for uncertain and imprecise population models. Also the work on automatic moment-closure analysis of population CTMC models by Feng, Hillston and Galpin [18] forms a bridge between work in WP3 and WP1 as it contributes a numerical technique to analyse the evolution of the first-order, the second-order, and the second-order joint moments of an arbitrary population CTMC model in an ODE based setting.

WP2 Collective Adaptive Behaviour in Space. There are several relationships to the work in WP2. In particular, the work by Feng and Hillston in which a technique is presented with which populations in large CAS models can be identified that do not have a significant impact on the target populations (i.e. those populations that have been selected for analysis), and thus can be removed from the model, reducing its size considerably. This approach may also be a promising technique when spatial aspects of CAS are considered. This is in part also shown in the bike sharing case study on which this technique is applied. Furthermore, it is likely that some of the other ODE reduction techniques may be applied to models that include spatial aspects. This is part of future work.

WP4 Language and Design Methodology. A number of the techniques presented in this deliverable can be readily extended to (a subset) of the CARMA language. In particular the work by Feng and Hillston mentioned earlier and that on automatic moment-closure analysis for population CTMC models.

WP5 Model Validation and Tool Support. For what concerns tool support, the reduction methods developed for chemical reaction networks have been implemented in a prototype tool that is described in more detail

in Sect. 4 of Deliverable 5.2. In the same section also a description of the tool for model reduction of PALOMA[5] models based on moment closure developed in [18] can be found, as well as a description of the FlyFast model checker based on the on-the-fly fast mean field model checking technique that was described in Deliverable 3.1 of WP3. The latter two tools have been implemented as a prototype on the Eclipse platform.

Roadmap for the final reporting period concerning Task 3.2 and Task 3.3. For the final reporting period of the project below we briefly outline the research we plan to undertake concerning Task 3.2 and Task 3.3. A roadmap for Task 3.1 of WP3 was already provided in Deliverable 3.1 of WP3 and will be further addressed in the forthcoming Internal Report that will appear in month 36.

For what concerns Task 3.2 we plan to focus on:

- The exploration and exploitation of the model reduction technique based on the identification of populations with insignificant impact on the selected target populations under analysis developed by Feng and Hillston. In particular its application in combination with statistical model checking. Also the application of ODE reduction techniques in combination with fluid model checking will be explored.
- Further work on CRN bisimulation. Differential bisimulation and forward CRN bisimulation are sufficient conditions for aggregation. It would be interesting to weaken their assumptions in order to be able to obtain coarser aggregates. For differential bisimulation, a possible strategy would be to give up compositionality, allowing for instance two processes not be distinguished if they perform different actions that have nonetheless the same effect on the underlying ODE system. It would be interesting to extend these to other popular kinetic mechanisms in systems biology, for instance Hill and Michaelis-Menten kinetics, because systems biology provides one with a rich repository of realistic large-scale models which can be used to study the effectiveness and scalability of the proposed reduction techniques.
- We plan to investigate other techniques to obtain tighter bounds for differential hulls. The developments on uncertain ODE models developed in WP1 and reported in Deliverable D1.2 appear to be promising in this respect.
- The investigation and development of abstraction techniques at the level of the CTMC semantics.

For what concerns Task 3.3 we plan to focus on:

- Establishing a subset of the modal μ-calculus that is preserved by (coherent) branching feature bisimulation (i.e., which properties are preserved by model reduction), which might lead to a feature-oriented modal μ-calculus.
- Considering further behavioural relations for VPA (e.g., a notion of variant bisimulation) and provide logical characterisations of all such relations to study model-checking properties of VPA models.
- Strengthening the integration of variability analysis (by multi-objective optimisation) of system configurations with performance analysis (by mean field model checking) of behavioural models. In particular we are aiming to use the outcome of performance analyses as input for variability modelling, in order to come to an automatic decision support system for the initial design of a bike-sharing system, as well as its successive adaptations and reconfigurations, taking both qualitative and performance aspects into consideration.
- Improving the QFLan implementation so that it scales better. Replace the current DTMC semantics with a CTMC one, in order to enrich the language with the notion of execution time. Evaluate the performance of the QFLan implementation, which includes applying it to a larger number of examples.

Where possible, we will consider the extension of the techniques to (a subset) of the CARMA language.

[5]The techniques will be made available for CARMA once a suitable dialect of CARMA is identified.

5 Acknowledgements

We would like to thank the Advisory Board of the QUANTICOL project for their valuable suggestions and Marco Bertini from PisaMo S.p.A. for fruitful discussions and sharing with us his knowledge on the Pisa bike sharing system.

References concerning publications of the Quanticol project for the reporting period

[1] D. Bacciu, S. Gnesi, and L. Semini. "Using a Machine Learning Approach to Implement and Evaluate Product Line Features". In: *WWV*. Ed. by M. H. ter Beek and A. Lluch Lafuente. Vol. 188. EPTCS. 2015, pp. 75–83. DOI: 10.4204/EPTCS.188.8.

[2] M. H. ter Beek, A. Fantechi, and S. Gnesi. "Applying the Product Lines paradigm to the quantitative analysis of Collective Adaptive Systems". In: *SPLC*. ACM, 2015, pp. 321–326. DOI: 10.1145/2791060.2791100.

[3] M. H. ter Beek, A. Fantechi, and S. Gnesi. "Challenges in Modelling and Analyzing Quantitative Aspects of Bike-Sharing Systems". In: *ISoLA*. Ed. by T. Margaria and B. Steffen. Vol. 8802. LNCS. Springer, 2014, pp. 351–367. DOI: 10.1007/978-3-662-45234-9_25.

[4] M. H. ter Beek, S. Gnesi, and F. Mazzanti. "Model Checking Value-Passing Modal Specifications". In: *PSI*. Ed. by A. Voronkov and I. Virbitskaite. Vol. 8974. LNCS. Springer, 2015, pp. 304–319. DOI: http://dx.doi.org/10.1007/978-3-662-46823-4_25.

[5] M. H. ter Beek and F. Mazzanti. "VMC: Recent Advances and Challenges Ahead". In: *SPLC*. Vol. 2. ACM, 2014, pp. 70–77. DOI: 10.1145/2647908.2655969.

[6] M. H. ter Beek and E. de Vink. "Using mCRL2 for the Analysis of Software Product Lines". In: *FormaliSE*. Ed. by S. Gnesi and N. Plat. IEEE, 2014, pp. 31–37. DOI: 10.1145/2593489.2593493.

[7] M. H. ter Beek and E. de Vink. "Software Product Line Analysis with mCRL2". In: *SPLC*. Vol. 2. ACM, 2014, pp. 78–85. DOI: 10.1145/2647908.2655970.

[8] M. H. ter Beek and E. de Vink. "Towards Modular Verification of Software Product Lines with mCRL2". In: *ISoLA*. Ed. by T. Margaria and B. Steffen. Vol. 8802. LNCS. Springer, 2014, pp. 368–385. DOI: 10.1007/978-3-662-45234-9_26.

[9] M. H. ter Beek, F. Damiani, S. Gnesi, F. Mazzanti, and L. Paolini. "From Featured Transition Systems to Modal Transition Systems with Variability Constraints". In: *SEFM*. Ed. by R. Calinescu and B. Rumpe. Vol. 9276. LNCS. Springer, 2015, pp. 344–359. DOI: 10.1007/978-3-319-22969-0_24.

[10] M. H. ter Beek, A. Legay, A. Lluch Lafuente, and A. Vandin. "Quantitative Analysis of Probabilistic Models of Software Product Lines with Statistical Model Checking". In: *FMSPLE*. Ed. by J. Atlee and S. Gnesi. Vol. 182. EPTCS. 2015, pp. 56–70. DOI: 10.4204/EPTCS.182.5.

[11] M. H. ter Beek, A. Fantechi, S. Gnesi, and F. Mazzanti. "Using FMC for Family-based Analysis of Software Product Lines". In: *SPLC*. ACM, 2015, pp. 432–439. DOI: 10.1145/2791060.2791118.

[12] M. H. ter Beek, A. Fantechi, S. Gnesi, and F. Mazzanti. *Modelling and Analysing Variability in Product Families: Model Checking of Modal Transition Systems*. Submitted for publication. 2015.

[13] M. H. ter Beek, A. Legay, A. Lluch Lafuente, and A. Vandin. "Statistical Analysis of Probabilistic Models of Software Product Lines with Quantitative Constraints". In: *SPLC*. ACM, 2015, pp. 11–15. DOI: 10.1145/2791060.2791087.

[14] M. H. ter Beek, S. Gnesi, D. Latella, and M. Massink. "Towards Automatic Decision Support for Bike-Sharing System Design". In: *SEFM collocated workshops*. Ed. by D. Bianculli. LNCS. To appear. Springer, 2015.

[15] T. Belder, M. H. ter Beek, and E. de Vink. "Coherent branching feature bisimulation". In: *FMSPLE*. Ed. by J. Atlee and S. Gnesi. Vol. 182. EPTCS. 2015, pp. 14–30. DOI: 10.4204/EPTCS.182.2.

[16] L. Cardelli, M. Tribastone, M. Tschaikowski, and A. Vandin. "Forward and Backward Bisimulations for Chemical Reaction Networks". In: *26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015*. Ed. by L. Aceto and D. de Frutos-Escrig. Vol. 42. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pp. 226–239. DOI: 10.4230/LIPIcs.CONCUR.2015.226. URL: http://dx.doi.org/10.4230/LIPIcs.CONCUR.2015.226.

[17] C. Feng and J. Hillston. "Speed-up of Stochastic Simulation of PCTMC Models by Statistical Model Reduction". In: *EPEW*. Vol. 9272. LNCS. Springer, 2015, pp. 291–305.

[18] C. Feng, J. Hillston, and V. Galpin. *Automatic Moment-closure Approximation of Spatially Distributed Collective Adaptive Systems*. Submitted for publication. 2015.

[19] G. Iacobelli, R. De Nicola, and M. Tribastone. "Dimming Relations for the Efficient Analysis of Concurrent Systems via Action Abstraction". In: *FORTE*. Vol. 8641. LNCS. Springer, 2014, pp. 216–231. URL: https://dl.dropboxusercontent.com/u/13100903/papers/forte2014.pdf.

[20] G. Iacobelli, M. Tribastone, and A. Vandin. "Differential Bisimulation for a Markovian Process Algebra". In: *Mathematical Foundations of Computer Science*. To appear. Quanticol technical report available at http://milner.inf.ed.ac.uk/wiki/files/W232G9A7/mfcs2015ExtendedTRpdf.html. 2015. URL: http://milner.inf.ed.ac.uk/wiki/files/W232G9A7/mfcs2015ExtendedTRpdf.html.

[21] M. Kowal, I. Schaefer, and M. Tribastone. "Family-Based Performance Analysis of Variant-Rich Software Systems". In: *FASE*. Vol. 8411. LNCS. Springer, 2014, 94–108. . URL: https://dl.dropboxusercontent.com/u/13100903/papers/fase2014.pdf.

[22] M. Tribastone. "Behavioral Relations in a Process Algebra for Variants". In: *SPLC*. IEEE, 2014, pp. 82–91. URL: https://dl.dropboxusercontent.com/u/13100903/papers/splc2014.pdf.

[23] M. Tschaikowski and M. Tribastone. "A unified framework for differential aggregations in Markovian process algebra". In: *J. Logic Algebra. Methods Program.* 84.2 (2015), pp. 238–258. ISSN: 2352-2208. DOI: http://dx.doi.org/10.1016/j.jlamp.2014.10.004. URL: https://dl.dropboxusercontent.com/u/13100903/papers/jlamp15.pdf.

[24] M. Tschaikowski and M. Tribastone. "Approximate reduction of heterogenous nonlinear models with differential hulls". In: *IEEE TAC* (2015). Available online. DOI: http://dx.doi.org/10.1109/TAC.2015.2457172.

[25] M. Tschaikowski and M. Tribastone. "Extended Differential Aggregations in Process Algebra for Performance and Biology". In: *QAPL*. Vol. 154. EPTCS. 2014, 34–47. . URL: https://dl.dropboxusercontent.com/u/13100903/papers/qapl2014.pdf.

References

[26] A. Antonik, M. Huth, K. Larsen, U. Nyman, and A. Wąsowski. "20 Years of Modal and Mixed Specifications". In: *Bulletin EATCS* 95 (2008), pp. 94–129.

[27] S. Apel, C. Kästner, A. Grösslinger, and C. Lengauer. "Type Safety for Feature-Oriented Product Lines". In: *ASE* 17.3 (2010), pp. 251–300.

[28] P. Asirelli, M. H. ter Beek, A. Fantechi, and S. Gnesi. "A Logical Framework to Deal with Variability". In: *IFM*. Ed. by D. Méry and S. Merz. Vol. 6396. LNCS. Springer, 2010, pp. 43–58. DOI: 10.1007/978-3-642-16265-7_5.

[29] P. Asirelli, M. H. ter Beek, A. Fantechi, and S. Gnesi. "Formal Description of Variability in Product Families". In: *SPLC*. Ed. by E. S. de Almeida, T. Kishi, C. Schwanninger, I. John, and K. Schmid. IEEE, 2011, pp. 130–139. DOI: 10.1109/SPLC.2011.34.

[30] K. Bąk, Z. Diskin, M. Antkiewicz, K. Czarnecki, and A. Wąsowski. "Clafer: unifying class and feature modeling". In: *Softw. Syst. Model.* (2015). DOI: 10.1007/s10270-014-0441-1.

[31] D. Batory. "Feature Models, Grammars, and Propositional Formulas". In: *SPLC*. Ed. by J. Obbink and K. Pohl. Vol. 3714. LNCS. Springer, 2005, pp. 7–20. DOI: 10.1007/11554844_3.

[32] M. H. ter Beek, S. Gnesi, and F. Mazzanti. "From EU Projects to a Family of Model Checkers: From Kandinsky to KandISTI". In: *Software, Services and Systems*. Ed. by R. De Nicola and R. Hennicker. Vol. 8950. LNCS. Springer, 2015, pp. 312–328. DOI: 10.1007/978-3-319-15545-6_20.

[33] M. H. ter Beek, A. Lluch Lafuente, and M. Petrocchi. "Combining Declarative and Procedural Views in the Specification and Analysis of Product Families". In: *SPLC*. Vol. 2. ACM, 2013, pp. 10–17. DOI: 10.1145/2499777.2500722.

[34] M. H. ter Beek, F. Mazzanti, and A. Sulova. "VMC: A Tool for Product Variability Analysis". In: *FM*. Ed. by D. Giannakopoulou and D. Méry. Vol. 7436. LNCS. Springer, 2012, pp. 450–454. DOI: 10.1007/978-3-642-32759-9_36.

[35] M. H. ter Beek, A. Fantechi, S. Gnesi, and F. Mazzanti. "A state/event-based model-checking approach for the analysis of abstract system properties". In: *Sci. Comput. Program.* 76.2 (2011), pp. 119–135. DOI: 10.1016/j.scico.2010.07.002.

[36] D. Benavides, S. Segura, and A. Ruiz-Cortés. "Automated Analysis of Feature Models 20 Years Later: a Literature Review". In: *Inf. Syst.* 35.6 (2010). DOI: 10.1016/j.is.2010.01.001.

[37] L. Bortolussi. "Stochastic Concurrent Constraint Programming". In: *ENTCS* 164 (3 2006), pp. 65–80. DOI: 10.1016/j.entcs.2006.07.012.

[38] P. Buchholz. "Exact and Ordinary Lumpability in Finite Markov Chains". In: *J. Appl. Probab.* 31.1 (1994), pp. 59–75. ISSN: 00219002.

[39] M. Buscemi and U. Montanari. "CC-Pi: A Constraint-Based Language for Specifying Service Level Agreements". In: *ESOP*. Ed. by R. De Nicola. Vol. 4421. LNCS. Springer, 2007, pp. 18–32. DOI: 10.1007/978-3-540-71316-6_3.

[40] A. Classen, M. Cordy, P. Schobbens, P. Heymans, A. Legay, and J. Raskin. "Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking". In: *IEEE TSE* 39.8 (2013), pp. 1069–1089. DOI: 10.1109/TSE.2012.86.

[41] A. Classen, M. Cordy, P. Heymans, A. Legay, and P. Schobbens. "Model checking software product lines with SNIP". In: *STTT* 14.5 (2012), pp. 589–612. DOI: 10.1007/s10009-012-0234-1.

[42] A. Classen, P. Heymans, P. Schobbens, A. Legay, and J. Raskin. "Model Checking Lots of Systems: Efficient Verification of Temporal Properties in Software Product Lines". In: *ICSE*. ACM, 2010, pp. 335–344. DOI: 10.1145/1806799.1806850.

[43] P. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley, 2002.

[44] M. Cordy, A. Classen, P. Heymans, P. Schobbens, and A. Legay. "ProVeLines: a product line of verifiers for software product lines". In: *SPLC*. Vol. 2. ACM, 2013, pp. 141–146. DOI: 10.1145/2499777.2499781.

[45] S. Cranen, J. Groote, J. Keiren, F. Stappers, E. de Vink, W. Wesselink, and T. Willemse. "An Overview of the mCRL2 Toolset and Its Recent Advances". In: *TACAS*. Ed. by N. Piterman and S. Smolka. Vol. 7795. LNCS. Springer, 2013, pp. 199–213. DOI: 10.1007/978-3-642-36742-7_15.

[46] F. Damiani and I. Schaefer. "Family-Based Analysis of Type Safety for Delta-Oriented Software Product Lines". In: *ISoLA*. Ed. by T. Margaria and B. Steffen. Vol. 7609. LNCS. Springer, 2012, pp. 193–207. DOI: 10.1007/978-3-642-34026-0_15.

[47] V. Danos and C. Laneve. "Formal molecular biology". In: *Theoret. Comput. Sci.* 325.1 (2004), pp. 69–110. ISSN: 0304-3975. DOI: http://dx.doi.org/10.1016/j.tcs.2004.03.065.

[48] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. "Abstracting the Differential Semantics of Rule-Based Models: Exact and Automated Model Reduction". In: *LICS*. IEEE, 2010, pp. 362–381. DOI: 10.1109/LICS.2010.44. URL: http://dx.doi.org/10.1109/LICS.2010.44.

[49] R. De Nicola and F. Vaandrager. "Action versus State based Logics for Transition Systems". In: *Semantics of Systems of Concurrent Processes*. Ed. by I. Guessarian. Vol. 469. LNCS. Springer, 1990, pp. 407–419. DOI: 10.1007/3-540-53479-2_17.

[50] R. De Nicola and F. Vaandrager. "Three Logics for Branching Bisimulation". In: *J. ACM* 42.2 (1995), pp. 458–487. DOI: 10.1109/LICS.1990.113739.

[51] R. De Nicola, A. Fantechi, S. Gnesi, and G. Ristori. "An Action Based Framework for Verifying Logical and Behavioural Properties of Concurrent Systems". In: *CAV*. Ed. by K. Larsen and A. Skou. Vol. 575. LNCS. Springer, 1991, pp. 37–47. DOI: 10.1007/3-540-55179-4_5.

[52] B. Delaware, W. Cook, and D. Batory. "A Machine-Checked Model of Safe Composition". In: *FOAL*. ACM, 2009, pp. 31–35.

[53] A. Fantechi and S. Gnesi. "Formal Modeling for Product Families Engineering". In: *SPLC*. IEEE, 2008, pp. 193–202. DOI: 10.1109/SPLC.2008.45.

[54] A. Fantechi, S. Gnesi, A. Lapadula, F. Mazzanti, R. Pugliese, and F. Tiezzi. "A logical verification methodology for service-oriented computing". In: *ACM TOSEM* 21.3 (2012), 16:1–16:46. DOI: 10.1145/2211616.2211619.

[55] C. Feng and J. Hillston. "PALOMA: A Process Algebra for Located Markovian Agents". In: *QEST*. Vol. 8657. LNCS. Springer, 2014, pp. 265–280. DOI: 10.1007/978-3-319-10696-0_22.

[56] C. Fidge. *A Comparative Introduction to CSP, CCS and LOTOS*. Tech. rep. 93-24. Software Verification Research Centre, University of Queensland, 1994.

[57] D. Fischbein, S. Uchitel, and V. Braberman. "A foundation for behavioural conformance in software product line architectures". In: *ROSATEA*. Ed. by R. Hierons and H. Muccini. ACM, 2006, pp. 39–48. DOI: 10.1145/1147249.1147254.

[58] S. Gnesi and F. Mazzanti. "On the Fly Verification of Networks of Automata". In: *PDPTA*. Ed. by H. Arabnia. CSREA, 1999, pp. 1040–1046.

[59] J. Groote and M. Mousavi. *Modeling and Analysis of Communicating Systems*. The MIT Press, 2014.

[60] J. Groote and F. Vaandrager. "An efficient algorithm for branching bisimulation and stuttering equivalence". In: *ICALP*. Ed. by M. Paterson. Vol. 443. LNCS. Springer, 1990, pp. 626–638. DOI: 10.1007/BFb0032063.

[61] A. Haber, T. Kutz, H. Rendel, B. Rumpe, and I. Schaefer. "Delta-oriented architectural variability using MontiCore". In: *ECSA*. Essen, Germany, 2011, 6:1–6:10. ISBN: 978-1-4503-0618-8.

[62] E. Haghverdi, P. Tabuada, and G. Pappas. "Bisimulation relations for dynamical, control, and hybrid systems". In: *Theoret. Comput. Sci.* 342.2–3 (2005), pp. 229–261. ISSN: 0304-3975. DOI: http://dx.doi.org/10.1016/j.tcs.2005.03.045. URL: http://www.sciencedirect.com/science/article/pii/S0304397505001660.

[63] M. Huhns and M. Singh. "Service-oriented computing: key concepts and principles". In: *Internet Computing, IEEE* 9.1 (2005), pp. 75–81. ISSN: 1089-7801. DOI: 10.1109/MIC.2005.21.

[64] K. Larsen, U. Nyman, and A. Wąsowski. "Modal I/O Automata for Interface and Product Line Theories". In: *ESOP*. Ed. by R. De Nicola. Vol. 4421. LNCS. Springer, 2007, pp. 64–79. DOI: 10.1007/978-3-540-71316-6_6.

[65] K. Larsen and B. Thomsen. "A Modal Process Logic". In: *LICS*. IEEE, 1988, pp. 203–210. DOI: 10.1109/LICS.1988.5119.

[66] D. Latella, M. Loreti, and M. Massink. "On-the-fly Fast Mean-Field Model-Checking". In: *TGC*. Ed. by M. Abadi and A. Lluch Lafuente. Vol. 8358. LNCS. Springer, 2014, pp. 297–314. DOI: 10.1007/978-3-319-05119-2_17.

[67] D. Latella, M. Loreti, and M. Massink. "On-the-fly PCTL fast mean-field model-checking for self-organising coordination". In: *Sci. Comput. Program.* (2015). Accepted. A preliminary version is available as QUANTICOL TR-QC-01-2013.

[68] K. Lauenroth, K. Pohl, and S. Töhning. "Model Checking of Domain Artifacts in Product Line Engineering". In: *ASE*. 2009, pp. 269–280.

[69] T. Lu and C. Law. "A directed relation graph method for mechanism reduction". In: *Proceedings of the Combustion Institute* 30.1 (2005), pp. 1333–1341.

[70] M. Mendonça, M. Branco, and D. Cowan. "S.P.L.O.T.: software product lines online tools". In: *OOPSLA'09*. Ed. by S. Arora and G. Leavens. ACM, 2009, pp. 761–762. DOI: 10.1145/1639950.1640002.

[71] L. de Moura and N. Bjørner. "Z3: An Efficient SMT Solver". In: *TACAS*. Ed. by C. Ramakrishnan and J. Rehof. Vol. 4963. LNCS. Springer, 2008, pp. 337–340. DOI: 10.1007/978-3-540-78800-3_24.

[72] A. Murashkin, M. Antkiewicz, D. Rayside, and K. Czarnecki. "Visualization and exploration of optimal variants in product line engineering". In: *SPLC*. ACM, 2013, pp. 111–115. DOI: 10.1145/2491627.2491647.

[73] G. Pappas. "Bisimilar linear systems". In: *Automatica* 39.12 (2003), pp. 2035–2047.

[74] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, 2005.

[75] V. Saraswat and M. Rinard. "Concurrent Constraint Programming". In: *POPL*. Ed. by F. Allen. ACM, 1990, pp. 232–245. DOI: 10.1145/96709.96733.

[76] I. Schaefer. "Variability Modelling for Model-Driven Development of Software Product Lines". In: *VaMoS*. 2010, pp. 85–92.

[77] I. Schaefer, L. Bettini, F. Damiani, and N. Tanzarella. "Delta-oriented programming of software product lines". In: *SPLC*. Springer, 2010, pp. 77–91. DOI: 10.1007/978-3-642-15579-6_6.

[78] A. J. van der Schaft. "Equivalence of dynamical systems by bisimulation". In: *IEEE TAC* 49 (2004).

[79] P. Schobbens, P. Heymans, and J. Trigaux. "Feature Diagrams: A Survey and a Formal Semantics". In: *RE*. IEEE, 2006, pp. 136–145. DOI: 10.1109/RE.2006.23.

[80] S. Sebastio and A. Vandin. "MultiVeStA: Statistical Model Checking for Discrete Event Simulators". In: *ValueTools*. Ed. by A. Horvath, P. Buchholz, V. Cortellessa, L. Muscariello, and M. Squillante. ACM, 2013, pp. 310–315. DOI: 10.4108/icst.valuetools.2013.254377.

[81] A. Singh and J. Hespanha. "Lognormal Moment Closures for Biochemical Reactions". In: *CDC*. IEEE, 2006, pp. 2063–2068.

[82] R. Singh, P. Shenoy, M. Natu, V. Sadaphal, and H. Vin. "Predico: A System for What-if Analysis in Complex Data Center Applications". In: *Middleware*. 2011, pp. 123–142.

[83] T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake. "A Classification and Survey of Analysis Strategies for Software Product Lines". In: *ACM Comput. Surv.* 47.1 (2014), 6:1–6:45. DOI: 10.1145/2580950.

[84] T. Thüm, C. Kästner, F. Benduhn, J. Meinicke, G. Saake, and T. Leich. "FeatureIDE: An extensible framework for feature-oriented software development". In: *Sci. Comput. Program.* 79 (2014), pp. 70–85. DOI: 10.1016/j.scico.2012.06.002.

[85] M. Tschaikowski and M. Tribastone. "Exact fluid lumpability for Markovian process algebra". In: *CONCUR*. LNCS. Springer, 2012, pp. 380–394. URL: https://dl.dropboxusercontent.com/u/13100903/papers/concur2012.pdf.

[86] B. Vogel-Heuser, D. Witsch, and U. Katzke. “Automatic code generation from a UML model to IEC 61131-3 and system configuration tools”. In: *ICCA*. IEEE, 2005, pp. 1034–1039.